MASTER THESIS

Study and Experimentation of Gender Bias in Co-reference Resolution

Author:
Felipe Alfaro

Advisor:
Marta Ruiz Costa-Jussà
Lluis Padro

A thesis submitted in fulfillment of the requirements for the degree of Master in Artificial Intelligence

Facultat D'Informàtica de Barcelona (UPC)
Facultat De Matemàtiques (UB)
Escola Tècnica Superior D'Enginyeria (URV)

June 18th, 2019

Abstract

Master in Artificial Intelligence

Study and Experimentation of Gender Bias in Co-reference Resolution

by Felipe Alfaro

Co-reference resolution is an important part of natural language understanding and it's been affected by the current corpora lacking in diversity. In the last year, Google AI Language's GAP dataset was created to provide an evaluation benchmark for how different models handle gender bias. The following project presents a series of experiments made with this dataset and different versions of the BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a state of the art NLP model released in 2018, it broke several records in well known language-based tasks and was praised within the community. The model is inspired by a lot of current techniques in the field, like semi-supervised learning and contextual embeddings. It uses a network of bidirectional transformers and it's trained using two semi-supervised tasks. These tasks are word prediction and sentence classification. BERT also released pre-trained models that can be adapted for specific tasks, the heavier part of the training is already done and all that is needed is fine-tuning with a labeled dataset.
It tackles the specific problem of co-reference resolution by working on a task created specifically for this dataset. It presents the implementation of two models for masked language modeling using pre-trained BERT adjusted to work for a classification problem. These two models take advantage of BERT's main capabilities, and they provide a way of solving co-reference resolution problems with BERT, which does not exist in BERT's original release. The proposed solutions are based on the word probabilities of the original BERT model, but using common English names to replace the original test names. Overall the model was successful predicting which entity was associated with each pronoun, but it had some failings when it came to predicting pronouns with no entity associated. Also, thanks to the parity of the training set, the model was also similarly successful in both female and male pronouns.

Acknowledgements

A huge thank you to Marta Ruiz Costa-Jussà for all of her help. Also thanks to Jose Adrián Rodriguez Fonollosa for his support and ideas. And to my friends and family who have helped me through this process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This project involves two elements that are gaining importance in the field. The first one of them is the element of gender in machine learning. This is a topic that has been discussed more as artificial intelligence becomes more prevalent in people's daily lives. As we know, any machine learning model is only as good as its training set. And, from the perspective of natural language, any corpora that exists carries with it the biases of the society in which it was made in. What happens when an artificial intelligence model trained with a bias has to make a decision with real implications? The creation of gender neutral datasets points to this being a very present problem in the industry. In 2018 Google AI Language's team created GAP, which they define as a gender-balanced dataset to address the gender bias in co-reference systems (Webster et al., 2018). 50% of its examples contain feminine pronouns, and the other 50% masculine pronouns. The task around this project is oriented towards building a pronoun resolution system that performs equally well regardless of pronoun gender.
Another motivation for this project is the rising popularity of new kinds of models for text analysis. In this case, we make use of the recently popularized BERT tool (Devlin et al., 2018). BERT is part of a what has been called the "ImageNet for NLP"[1] and it's expected to be as ubiquitous for NLP in the future as ImageNet is for computer vision today. BERT is a model trained for masked language modeling (LM) word prediction and sentence prediction using the transformer network (Vaswani et al., 2017a). BERT also provides a group of pre-trained models for different uses, of multiple languages and sizes. There are implementations for it in all sorts of tasks, including text classification, question answering, multiple choice question answering, sentence tagging, among others. BERT is gaining popularity quickly in natural language solutions, but before this shared-task appeared, we had no awareness of its implementation in co-reference resolution. For this task, we've used an implementation that takes advantage of the masked LM which BERT is trained for and uses it for a kind of task BERT is not specifically designed for.

1.1 Problem

The main problem presented in this project was first proposed as a task for the 1st ACL Workshop on Gender Bias for Natural Language Processing [2]. The Gendered Pronoun Resolution task is a natural language processing task whose objective is to build pronoun resolution systems that identifies the correct name a pronoun in a text refers to. It's called a co-reference resolution task. Co-reference resolution tackles the problem of different elements of a text that refer to the same thing, like for example

[1] http://ruder.io/nlp-imagenet/
[2] https://genderbiasnlp.talp.cat/

a pronoun and a noun, or multiple nouns that describe the same entity. The current task also has to deal with the problem of gender. As the GAP researchers point out (Webster et al., 2018), the biggest and most common datasets for co-reference resolution have a bias towards male entities. For example the OntoNotes dataset, which is used for some of the most popular models, only has a 25% female representation (Pradhan and Xue, 2009). This creates a problem, because any machine learning model is only as good as its training set. Biased training sets will create biased models, and this will have repercussions on any uses the model may have.

1.2 Task Description

The specifics of the co-reference resolution task are the following. Every entry of the dataset contains:

- Text of variable length, usually a few sentences
- A pronoun that is present on the text. The dataset has the word itself and its position on the text.
- The name A. Which is a name present once or more on the text. The dataset has the name itself and its position on the text. If there are multiple appearances it has the position of only one.
- The name B. Another name present once or more on the text. The dataset has the name itself and its position on the text. If there are multiple appearances it has the position of only one.

In every text the pronoun can reference one of three things, name A, name B or neither. For example lets take the following text, the pronoun is the **his** in the last sentence, and the names A and B are **Bilbo** and **Gandalf** respectively. In this case the pronoun clearly refers to Bilbo, so the correct answer is **A**.

Sixty years later, Gollum was captured by orcs, and taken to Mordor, where he was tortured into revealing the owner and location of the Ring; ***Bilbo*** *Baggins of the Shire. In the meantime,* ***Bilbo*** *had left the Shire to live in Rivendell, and upon the advice of* ***Gandalf*** *had (very reluctantly) given the Ring to* ***his*** *nephew, Frodo Baggins.*

Another example is in following text, the pronoun is she and the names A and B are **Jenee Welsh** and **Jennifer** respectively. The pronoun refers to Laura, but she's not one of the given names so the correct answer is **None**.

The final character was Laura, the dumb brunette, a model unaware of her physical attractiveness. Laura was the key character around which most of the show's situations revolved. Her caption would change every episode and formed the title of the episode, such as "and LAURA this week ***she****'s on a diet", "This Week She Wants to Be a Singer", "This Week She Travels", etc. The regular cast included Ken James as Mark, Gregory Ross as Bob, Gregory de Polnay as Jeremy,* ***Jenee Welsh*** *as* ***Jennifer*** *and Terry O'Neill as Tinto.*

1.3 Contribution

As one of the project's contributions, we propose two models for co-reference resolution using BERT that were successful for the task. Both models use BERT in a way that complements the way in which it was designed. We also propose our method of name replacement which was used in both models with great success.
This project had a successful participation in the Kaggle competition related to the

shared task [3], where it placed 46th out of 263 submissions. And also a short task paper was submitted and accepted by the workshop. This paper can be found in the appendices.

[3] https://www.kaggle.com/c/gendered-pronoun-resolution

Chapter 2

Theoretical Background

BERT is a model trained for masked language modeling (LM) word prediction and sentence prediction using a transformer network (Vaswani et al., 2017a). Its initials stand for **Bidirectional Encoder Representations from Transformers**. BERT has been growing quickly in popularity in the NLP space. In this section we will explain how it works on the inside, how it was trained and the multiple uses it has.

2.1 Transformer

Transformers are a type of component in complex Deep Learning models, they have become incredibly popular recently and they are fundamental to how BERT works. In this section we give a more detailed explanation of what constitutes a Transformer. There are multiple types of transformers, but for this document when we refer to one we are referring to the one used by BERT, which was described first in the paper by Vaswani et al. (2017b).
A transformer has an **encoder-decoder** structure. The **encoder** takes a symbol representation x (in this case text) and converts it to a continuous space z. Then, the **decoder** converts from this continuous space back to a symbol representation y. In the case of BERT they are both the same language but transformers are also commonly used for machine translation where x and y are different languages.

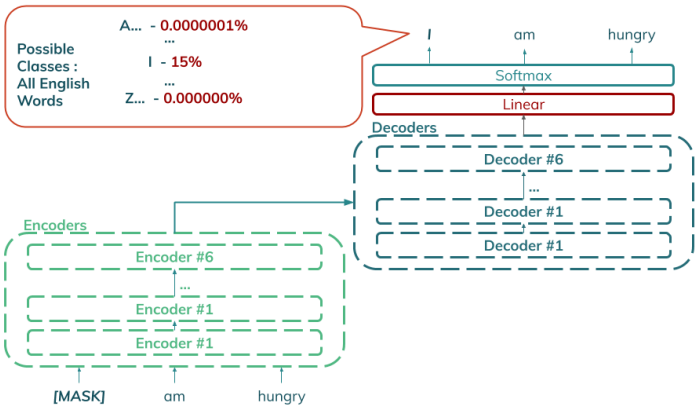

FIGURE 2.1: An overview of how a transformer works. Here the input text is *I am hungry* with the *I* masked. The output is correctly predicting the masked word.

2.1.1 Encoder

Each encoder is composed of two sub-layers, the first one is a multi-head self attention mechanism and the second is a feed forward neural network. Connecting them to each other and to the following layers are normalization sub-layers. All the sub-layers in the encoders have **residual connections** to each other. These are used to allow gradients to go through the sub-layers directly. On the right side of figure 2.2 is the representation of the encoder on a smaller scale with only 2 layers. In the original paper there are 6 layers of encoders as well as 6 of decoders (Vaswani et al., 2017b), while in BERT the number varies between 12 and 16 (Devlin et al., 2018).

2.1.2 Decoder

The decoder on the inside has the same sub-layers as the encoder, and it adds an additional one, which is multi-head attention that runs over output of the encoding layers. The decoder also has residual connections between all sub-layers (Vaswani et al., 2017b).

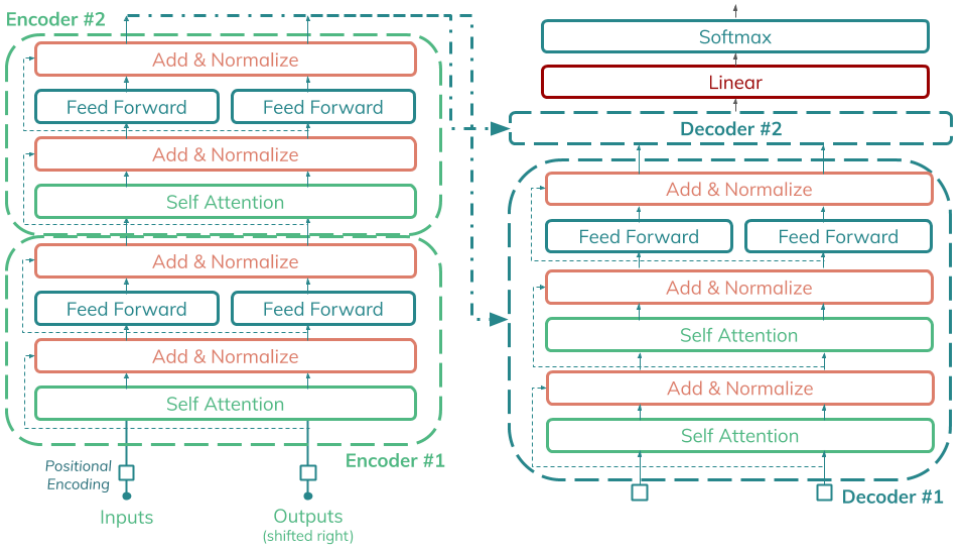

FIGURE 2.2: A small representation of a transformer with 2 encoding layers and 2 decoding layers.

2.1.3 Sub-Layers

Attention Sub-Layers

Attention sub-layers are used to sum all the input vectors using a weighted compatibility function that takes into account the whole output of the previous layer. As each word passes through the encoder, the self attention sub-layer allows the model to look at the entire input sequence to determine how much it should factor into the sum. This helps the model identify words that are related to each other and associate words that need to go together to give the text proper meaning. The transformer uses a specific kind of self attention sub-layer called **multi-head**. This kind of attention increases the model's ability to focus on different positions by splitting the embedding of the word into multiple heads, calculates the attention for them separately and then concatenates the results (Vaswani et al., 2017b).

Feed Forward Sub-Layers

Both the encoder and the decoder use feed forward sub-layers. These are all identical to each other with only the parameters changing. Each one consist of two linear transformations with RELU activation (Vaswani et al., 2017b).

Linear and Softmax Layer

The Linear layer is a fully connected neural network that projects the vector produced by the stack of decoders into a vector of the size of the model's vocabulary. This is called the **logits vector**. This vector is then passed through a softmax that transforms the values into probabilities, where each position in the vector represents the probability an element in the vocabulary. As shown in the top right corner of figure 2.1.

2.2 BERT's Architecture

One of the main advantages BERT has is that it's designed for bidirectional training, while most models are unidirectional (left to right or right to left). Unidirectional training is very limiting for models that require the whole context of a text, like question answering or sentence prediction, as each word can only see the ones that came before (Devlin et al., 2018).
BERT is trained in two different sizes, BERT BASE and BERT LARGE. The differences between them can be seen in table 2.1. The transformer blocks are the transformers explained in section 2.1, which can have between 12 and 16 self attention heads. In BERT as each token is passing through the transformer it has the knowledge of the previous token as well as the next one because of the model's bidirectionality. This gives it an advantage over models like ELMo or OpenAI GPT (Devlin et al., 2018).

	BASE	**LARGE**
Layers (Transformer Blocks)	12	24
Hidden Size	768	1024
Self Attention Heads	12	16
Total Parameters	110M	340M

TABLE 2.1: Differences between both BERT sizes

BERT is trained with two different tasks: Masked LM and Next Sentence Prediction.

1. **Masked LM :** The bidirectional approach causes a problem, which is that indirectly every token can "see itself". BERT fixes this by using masking, 15% of the tokens in each text are replaced. Most of the time they are replaced with the token *[MASK]*, but they can also be replaced with another random word or with the original word. This last case is done to avoid problems that could arise during fine-tuning because the token *[MASK]* would be absent. The task itself consists of guessing the original word that belonged in the place of this masked tokens (Devlin et al., 2018).

2. **Next Sentence Prediction :** The second task used for training is designed to help the model understand the relationships between two sentences. The objective is to predict if the second sentence B following a the first sentence A. For this, BERT uses its way of doing classification. Every entry has a tag at the beginning of the text (the *[CLS]* tag), the final results from this tag are then passed through a feed forward neural network that trains on two kinds of examples. When sentence B follows A is True and when it's false. BERT uses another a custom tag *[SEP]* to separate sentences in each of the texts, as well as positional encoders.

2.2.1 Input Representation

BERT uses a combination of 3 different layers for input representation, a token embeddings layer, a positional embeddings layer and a segment embeddings layer (fig 2.3).
The tokenization for the BERT uses what is known as Wordpiece segmentation, which is a data-driven method designed for balancing the vocabulary as well as the out of vocabulary words. Wordpieces are not equivalent to tokens, some tokens have to be split into multiple wordpieces according to their number of appearances in the data. This is how BERT has a vocabulary size of 30522 words while also very rarely encountering problems with "out of vocabulary" tokens (Wu et al., 2016). Each wordpiece is then represented in the **token embeddings layer** as a vector (Devlin et al., 2018).
The second layer of embeddings is called the **segment embeddings layer** and it's used for determining which part of the sequence each of the tokens is in. It's basically a mask with either 0s or 1s that tells the model if it's reading a token from part 1 of the sequence or part 2. For example, for multi-sentence classification (2.2.3) or multiple choice answering (2.2.3), the two parts of the text are placed in the same sequence in the form *[CLS] part1 [SEP] part2 [SEP]*. The mask for this example would be *[CLS] 0 0 ... 0 0 [SEP] 1 1 ... 1 1 [SEP]*. Finally BERT uses **position embeddings** to give context for each word's place in the sentence (Shukri H., 2019a). A word that appears twice in the same sentence would never have the same positional embedding. For example in the phrase *I think, therefore I am* the first and the second *I* have the same token embedding but can be distinguished from each other using their positional embeddings (Devlin et al., 2018).

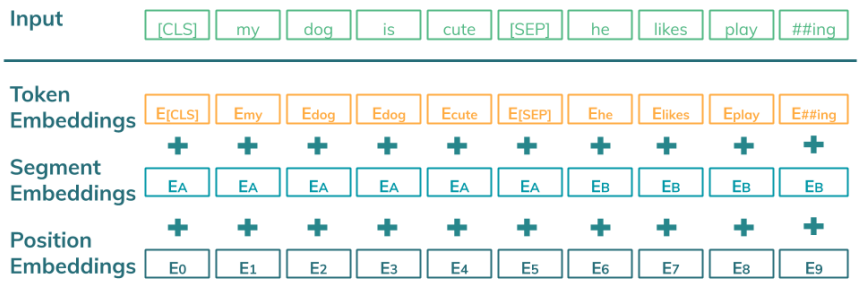

FIGURE 2.3: Graph of the three layers of embeddings used by BERT

2.2.2 BERT training

Google provides several BERT pre-trained models for several uses. These models eliminate the necessity of creating and training massive architectures for each new task. English BERT is trained with BooksCorpus (800 million words) and the English version of Wikipedia (2500 million words) (Devlin et al., 2018). To create each entry, two spans of text from the corpus were sampled. These are called sentences even if the length is not strictly the same as a sentence. The first of these sentence is labeled A and is given the A positional embedding and the second sentence gets the same with B. 50% of the time B is the actual next sentence that follows A and the rest of the time it is a random sentence. For the LM masking, 15% of tokens are masked after WordPiece tokenization has been performed, with no consideration of whether a WordPiece is a complete word or a segment of one. BERT is trained with 256 sequences for 1,000,000 steps. Adam is the chosen optimizer The training loss is calculating by adding the mean masked LM likelihood and mean next sentence prediction likelihood (Devlin et al., 2018).

2.2.3 BERT uses

BERT also provides implementations for different kinds of tasks, including text classification, question answering, multiple choice question answering, sentence tagging, among others (Devlin et al., 2018). Masked LM and Sentence prediction are mentioned in section 2.2.

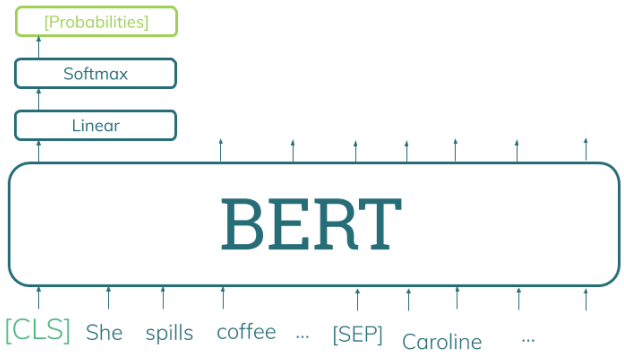

FIGURE 2.4: BERT model used for multi-sentence classification

BERT for text classification

BERT can be used for single sentence classification or multiple sentence classification. In both cases the *[CLS]* tag at the beginning of the sentence is used. The results from the class label are then passed through a feed forward neural network with a softmax layer to determine the probability of the text belonging to each class. This can be seen in figure 2.4.
For multi-sentence classification it follows the same model, with the difference that the text contains the *[SEP]* tag to denote when a sentence ends and another one begins. This model is used to understand and predict the relationship between two parts of the text. For example whether one text follows another one logically, or if

one sentences is the answer to a question. Note that we refer to sentences in the context of BERT but they could be texts or phrases of any length.

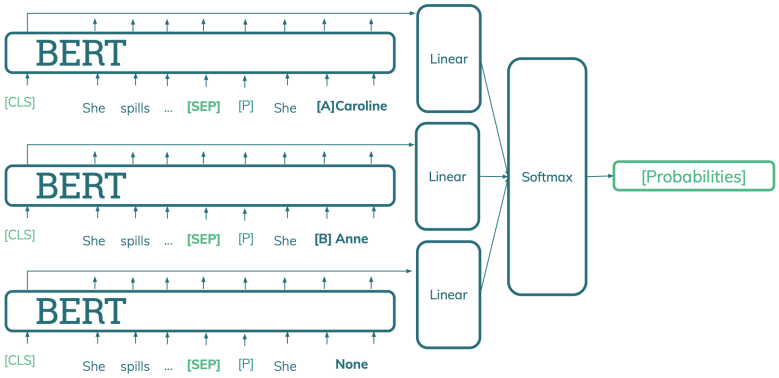

FIGURE 2.5: BERT model used for multiple choice question answering

BERT for Multiple Choice Question Answering

The model for multiple choice takes the concept from the multi-sentence classification and complements it. Every entry is composed of a specific series of elements. The context contains the text where the information is present as well as the question that is being asked. This is present before the *[SEP]* tag. Then, for each of the multiple choice answers, a separate entry is created. These entries have the answer after the *[SEP]* tag. This means for every question in the dataset we get N entries for BERT to process. Finally the results of the *[CLS]* tag are passed through a feed forward neural network, but in this case instead of trying to predict the class it's trying to predict whether the question/answer combination is True or False. Finally the results of all entries for each question are joined and passed through a softmax layer to determine the correct answer (Devlin et al., 2018).

Chapter 3

State of the Art

This project is based on the research in two topics: Co-reference Resolution and Gender Bias in NLP. The following chapter explains the recent discoveries made in those fields that made this project possible.

3.1 Co-reference Resolution

Co-reference resolution is a natural language processing task that determines when different elements in a text refer to the same real world entity. The sentence : *John told Sally she should come and watch him play the violin.* is a good example for this. In this sentence we have two named entities, *John* and *Sally*, as well as two pronouns, *him* and *she*. To fully understand the text it's important that we know which pronoun refers to which entity. Co-reference resolution is vital to tasks like machine translation and tools like chatbots.
Deep learning models have been gaining popularity in this space. Clark and Manning (2016) propose a deep learning model that gives out scores for mention pairs. Mention pairs are Every mention pair in the form (m_i, m_j) passes through two neural networks, both with 4 fully connected layers. The first one is trained to give a score for the probability of m_i being the antecedent of m_j. The second one is the score referring to m_j having no antecedent. So, for the example we had before with John and Sally, the pairs *(John, him)*, (John, she), (Sally, him) and (Sally, she) would be evaluated and given a score (Clark and Manning, 2016).
After the competition's other participants have also made contributions in the form of new models and approaches. Liu (2019) proposed a combination of a End to End model that scores each mention and the BERT classification model. This End to End model, originally proposed by Lee et al. (2017), uses a network composed of bidirectional LSTMs to calculate an antecedent score and a mention score for each mention pair. Liu (2019) takes the mention score for both of the names and combines this with the results of a pre-trained BERT classification model. An interesting fact about this last model is that it also uses a name replacement technique, although with slightly different rules, it only replaces the names in about 80% of the examples.
The competition winning model was proposed by Attree (2019) and it's a combination of fine-tuned BERT and a Evidence Pooling module. This module consists of a neural network that uses a self-attention mechanism and attention pooling to calculate a score for the combination of the text, the pronoun and both names. Then they concatenate this with the BERT results to obtain the final score. This solution had a 0.137 loss and placed 1st in the Kaggle competition, winning by a significant margin.

3.2 Gender Bias in NLP

The problem of gender bias in machine learning has been pointed out a lot recently. As the GAP researchers point out, the biggest and most common datasets for co-reference resolution have a bias towards male entities (Webster et al., 2018). For example the OntoNotes dataset, which is used for some of the most popular models, only has a 25% female representation (Pradhan and Xue, 2009). This creates a problem, because any machine learning model is only as good as its training set. Biased training sets will create biased models, and this will have repercussions on any uses the model may have. Any corpora will suffer from the social biases of the environment in which it's made. Those biases are then transferred to any machine learning that uses them. For example word embeddings made from a corpora with certain biases inherits those biases into it (Caliskan, Bryson, and Narayanan, 2016). Bolukbasi et al. (2016) studies how bias appears in word embeddings and proposes a way of fixing it. By finding the words that show the sharpest biases in embeddings, and nullifying them in the embedding space, as well as equalizing the distance between them and genders. Zhao et al. (2018) proposes GN-GloVe, an algorithm to generated gender neutral word embeddings. It takes Glove as a base and creates a "protective attribute". This protected attribute is reserved to its own space, allowing the rest of the embeddings to be created without it. This approach can be used to any attribute not just gender.

Chapter 4

Architecture

This chapter covers the architecture of both models designed for the task explained in section 1.2. We will refer to these models as model 1 and model 2, they both have the same base which is the BERT for Masked LM model. This section also explains the name replacement feature and how it was used to complement the models.

4.1 BERT for Masked LM

Several BERT implementations were tested for this task, in the end, the most successful ones ended up being adapted versions of the Masked LM task used to train the original BERT model itself. Both of these models rely on the fact that BERT is at its best when doing word prediction. These models were both submitted to the competition with similar success. BERT for Masked LM's main objective is to predict a word that has been masked in a sentence. For this task, that word will always be the pronoun whose referent we're trying to identify. This one pronoun gets replaced by the *[MASKED]* tag, the rest of the sentence is subjected to the different name change rules described in section 4.4.
The text is passed through a pre-trained BERT model. After the text has passed through BERT, the resulting sequence then passes through what is called the **masked language modeling head**. This consists of a feed forward neural network that returns, for every word in the sequence, an array the size of the entire vocabulary with the probability for every word. The array for our masked pronoun is extracted and then from that array, we get the probabilities of three different words. These three words are : the first replaced name (name 1), the second replaced name (name 2) and the word *none* for the case of having none (figure 4.1). The only fine-tuning made in training is done to the masked language modeling head.
This third case is the strangest one, because the word *none* would logically not appear in the sentence. Tests were made with the original pronoun as the third option instead. The results ended up being very similar albeit slightly worse, so the word none was kept instead. These cases where there is no true answer are the hardest ones for both of the models.

4.2 Model 1

In the first model, after the probabilities for each word are extracted, the rest is treated as a classification problem. An array is created with the probabilities of the 2 names and *none*. **[name 1, name 2, none]**, with each one representing the probability of a class in multi-class classification. It's passed through a softmax function to

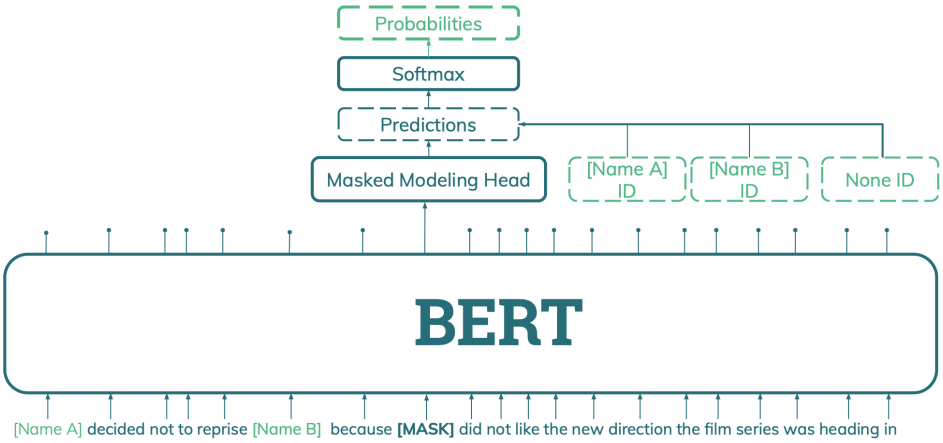

FIGURE 4.1: Model 1 representation.

adjust it to probabilities between 0 and 1 and then the log loss is calculated. A visual of this model can be seen in figure 4.1.

4.3 Model 2

The second model repeats the steps of model 1 but for two different texts. These texts are mostly the same except the replacement names *name 1* and *name* 2 have been switched (as explained in the section 4.4). It calculates the probabilities for each word for each text and then takes an average of both. Finally, it applies the softmax and calculates the loss with the average probability of each class across both texts. A visual of this model can be seen in figure 4.2.

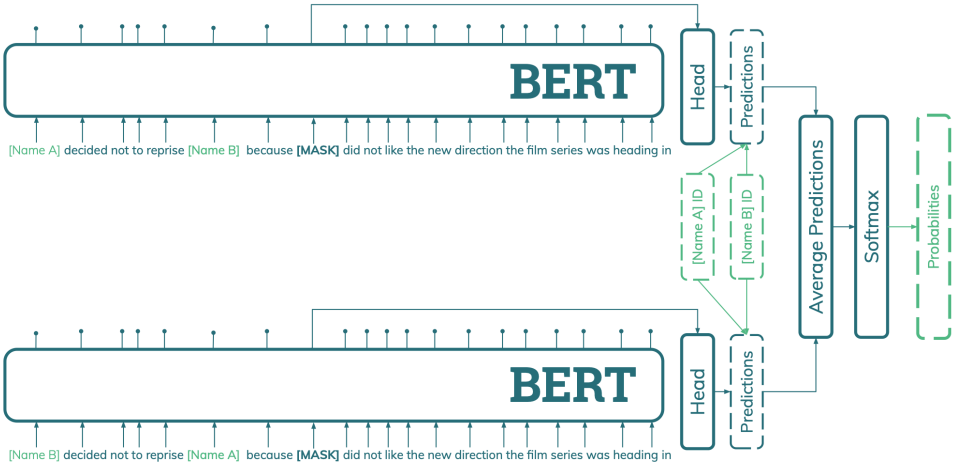

FIGURE 4.2: Model 2 representation.

4.4 Name Replacement

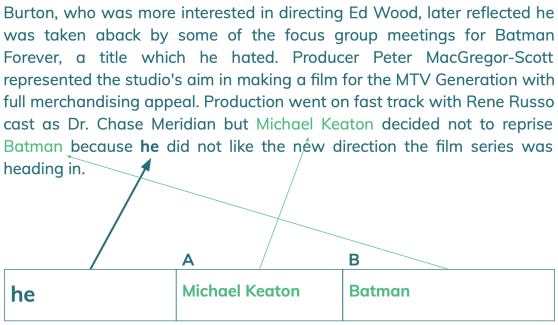

FIGURE 4.3: Example of a text present in the dataset and how the word replacement was done for the model 1.

The dataset contains only names of individuals who are featured in Wikipedia and some of these names are uncommon in the English language. Because of this, as part of the pre-processing for both models, these names are replaced. They are replaced with common English names in their respective genders[1]. If the pronoun is female, one of two common English female names are chosen, same thing for the male pronouns. In order to replace them in the text, the following set of rules are followed.

1. The names mentioned on the A and B columns are replaced.
2. Any other instances of the full name as it appears on the A/B columns are replaced.
3. If the name on the A/B column contains a first name and a last name. Instances of the first name are also replaced. Unless both entities share a first name, or the first name of one is contained within the other.
4. Both the name and the text are converted to lowercase

For example lets say we get the text:
"In the late 1980s Jones began working with Duran Duran on their live shows and then in the studio producing a B side single "This Is How A Road Gets Made", before being hired to record the album Liberty with producer Chris Kimsey.", A is *Jones* and B is *Chris Kimsey*. For the name replacement lets say we choose two common english names like **John** as **replacement A** and **Harry** as **replacement B**. The new text produced for model 1 (figure 4.1) would be something like:
*"in the late 1980s **john** began working with duran duran on their live shows and then in the studio producing a b side single "this is how a road gets made", before being hired to record the album liberty with producer **harry**."*
For model 2 (figure 4.2), the same text is treated to the name replacement twice to produce two different texts. The first of these texts is the same as the one used for model 1, the second one has replacement A and replacement B switched. So the second text our previous example we look like: *"in the late 1980s **harry** began working with duran duran on their live shows and then in the studio producing a b side single "this is how a road gets made", before being hired to record the album liberty with producer **john**."*

[1] https://www.ef.com/wwen/english-resources/english-names/

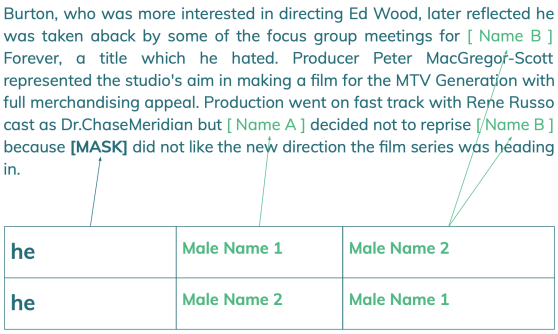

FIGURE 4.4: Example of a text present in the dataset and how the word replacement was done for the model 2.

This name replacement has two major benefits. First, the more common male and female names work better with BERT because they appear more in the corpus in which it is trained on. Secondly, when the wordpiece encoding splits certain words the tokenizer can be configured so that our chosen names are never split. So they are single tokens (and not multiple word pieces), which helps the way the model is implemented.

Both models (1 and 2 presented in the above section) use BERT for Masked LM prediction where the mask always covers a pronoun, and because the pronoun is a single token (not split into word pieces), it's more useful to compare the masked pronoun to both names, which are also both single tokens. The name John will always be a single token and a single token while another name might be 3 or more tokens. Because the chosen names are very common in the English language, BERT's previous training might contain biases towards one name or the other. This can be detrimental to this model where it has to compare between only 3 options. Our alternative to tackle this problem is the approach in model number 2, where the names are switched and then an average is made, to compensate for any possible bias.

Chapter 5

Experiments

This chapter describes the experiments performed to arrive at the final version of the models, as well as how the models performed in the competition, and in which areas did they excel and in which did they have the most problems.

5.1 Metrics

In order to measure the effectiveness of the models two metrics were chosen, the accuracy and the loss. These are very common metrics for classification problems and have a universal understanding. The log loss was chosen because it was also the official metric of the competition. This metric measures the precision of a classification model where the results are probabilities. The lower the probability of the correct result goes the more the loss increases, so the model's objective is to minimize the log loss. The other metric used is the accuracy, this is a well known metric for any classification model. It represents how many results were correctly classified. The reason why the log loss is more valuable as a metric for us is because it takes into account the confidence in each response. The accuracy just values true or false results, so a correct answer with a 0.9 probability has the same value as a correct answer with a 0.5 probability. The log loss takes into this into account making the confidence matter.

5.2 Experimental Framework

5.2.1 Task details

The objective of the task is that of a classification problem. Where the output for every entry is the probability of the pronoun referencing name A, name B or Neither. A full explanation of the task is in section 1.2.

5.2.2 Data

The GAP dataset created by Google AI Language was the dataset used for this task (Webster et al., 2018). This dataset consists of 8908 co-reference labeled pairs sampled from Wikipedia, also it's split perfectly between male and female representation. Each entry of the dataset consists of a short text, a pronoun that is present in the text and its offset and two different names (name A and name B) also present in the text. The pronoun refers to one of these two names and in some cases, none of them. The GAP dataset doesn't contain any neutral pronouns such as *it* or *they*.
For the two different stages of the competition different datasets were used.

- For **Stage 1** the data used for the submission is the same as the development set available in the GAP repository. The dataset used for training is the combination of the GAP validation and GAP testing sets from the repository.
- For **Stage 2** the data used for submission was only available through Kaggle[1] and the correct labels have yet to be released, so we can only analyze the final log loss of each of the models. This testing set has a total of 12359 rows, with 6499 male pronouns and 5860 female ones. For training, a combination of the GAP development, testing and validation sets was used. And, as all the GAP data, it is evenly distributed between genders.

The distributions of all the datasets are shown in table 5.1. It can be seen that in all cases, the *None* option has the least support by a large margin. This, added to the fact that the model naturally is better suited to identifying names rather than the absence of them, had a negative effect on the results.

	Stage 1		**Stage 2**
	Training Set	**Testing Set**	**Training Set**
Name A	1105	874	1979
Name B	1060	925	1985
None	289	201	490

TABLE 5.1: Dataset distribution for the datasets of stages 1 and 2.

5.2.3 Training details

For the BERT pre-trained weights, several models were tested. BERT base is the one that produced the best results. BERT large had great results in a lot of other implementations, but in this model it produced worse results while consuming much more resources and having a longer training time. During the experiments the model had an overfitting problem, so the learning rate was tuned as well as a warm up percentage was introduced. As table 5.3 shows, the optimal learning rate was $3e-5$ while the optimal with a 20% warm up. The length of the sequences is set at 256, where it fits almost every text without issues. For texts too big, the text is truncated depending on the offsets of each of the elements in order to not eliminate any of the names or the pronoun.

Parameter	**Value**
Optimizer	Adam
Vocabulary Size	28996
Dropout	0.1
Sequence Length	256
Batch Size	32
Learning Rate	$3e-5$
Warm Up	20%
Steps	Stage 1: 81 | Stage 2: 148
Epochs	1
Gradient Accumulation Steps	5

TABLE 5.2: Hyperparameters for the model training

[1] https://www.kaggle.com/c/gendered-pronoun-resolution/overview

The training was performed in a server with an Intel Dual Core processor and Nvidia Titan X GPUs, with approximately 32GB of memory. The run time varies a lot depending on the model. The average run time on the stage 1 dataset for model 1 is from 1 to 2 hours while for model 2 it has a run time of about 4 hours. For the training set for stage 2, the duration was 4 hours 37 minutes for model 1 and 8 hours 42 minutes for model 2. The final list of hyperparameters is in table 5.2.

		Accuracy		Loss	
Learning Rate	**Warmup**	**mean**	**min**	**mean**	**min**
0.00003	**0.0**	0.840167	0.8315	0.519565	0.454253
	0.2	0.844444	0.8340	0.502667	0.442313
0.00004	**0.0**	0.822389	0.7970	0.556491	0.473528
	0.2	0.834000	0.7925	0.530862	0.456223
0.00005	**0.1**	0.743500	0.7435	0.666750	0.666750
0.00006	**0.0**	0.756333	0.7040	0.630707	0.544841
	0.2	0.802278	0.7465	0.587041	0.497051

TABLE 5.3: Results of the tuning for both models. Minimum and average Loss and Accuracy across all the tuning experiments performed.

5.3 Competition Results

	Model 1	**Model 2**
Stage 1	0.44231	0.49607
Stage 2	0.31441	0.30151

TABLE 5.4: Results for both models across both stages of the competition

The official competition was made in Kaggle and it consisted of two stages. Stage 1 was made using Google's GAP development dataset while stage 2 was made with a new custom dataset created specifically for the competition. The correct classes for the stage 2 dataset have not been released to the public, so the analysis we can make about the differences between both stages is limited. Also both stages have different training sets for the models. Stage 1 uses the testing and validation parts of the GAP dataset while stage 2 uses the whole GAP dataset.
The stage 1 dataset is considerably smaller with 2000 rows compared to the 12359 rows of the stage 2 dataset. But, stage 2 does have on average smaller texts, which can be easier for the model to process and also don't suffer from truncating.

	Used in	**Entries**	**Average Length (tokens)**
GAP Testing + Validation	Stage 1 Training	2454	71.4625
GAP Development	Stage 1 Submission	2000	71.2035
GAP (union)	Stage 2 Training	4454	71.3462
Test Phase 2	Stage 2 Submission	12359	61.8033

TABLE 5.5: Different datasets used for training and for the Kaggle competition

In the official competition on Kaggle we placed 46th, with the model 2 having a loss around 0.301. As the results in table 5.5 show, the results of stage 2 were better than those of stage 1. This could be because the training set for the second stage was much larger. Also, unexpectedly, model 2, which had performed worse on the first stage was better in stage 2.

5.4 Model 1 vs Model 2

	Precision	Recall	F1	Support
A	0.83	0.87	0.85	874
B	0.88	0.88	0.88	925
None	0.64	0.52	0.57	201
Avg	**0.83**	**0.84**	**0.84**	**2000**

TABLE 5.6: Model 1 results for the testing stage 1.

	Precision	Recall	F1	Support
A	0.81	0.86	0.83	874
B	0.88	0.78	0.82	925
None	0.48	0.62	0.54	201
Avg	**0.81**	**0.80**	**0.80**	**2000**

TABLE 5.7: Model 2 results for the testing stage 1.

Overall, both models were successful in predicting the names. Their main failing was the predicting of results that were *None*. This is mainly because of two reasons, the training set has considerably less *None* results and also the model's approach of comparing to two names and the word none is not as well suited to *None* cases. *None* examples are also fewer in the submission sets, which is why the overall results are still positive.
As it can be seen on tables 5.6 and 5.7, model 1 had better results than model 2 on almost all of our metrics. Unfortunately we don't have the breakdown of the classes in submission 2, which means we don't know for certain what made model 2 more successful in this stage.

5.5 Masked LM vs Other BERT implementations

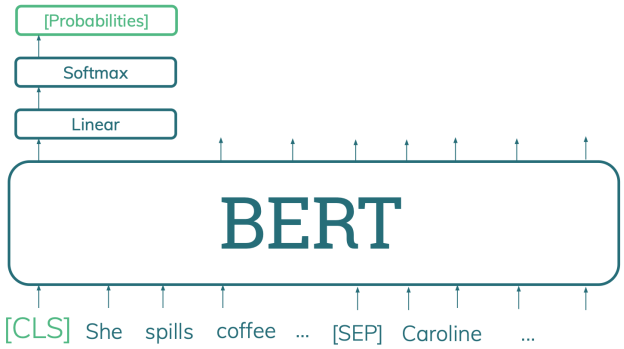

FIGURE 5.1: Model: BERT for text classification

As well as the Masked LM, three other BERT implementations were experimented with for the task. First, a text multi-class classification model (figure 5.1) where the *[CLS]* tag is placed at the beginning of every sentence, the text is passed

through a pre-trained BERT and then the result from this label is passed through a feed forward neural network. It's the default BERT text classification model where in our case the three classes are name A, name B and None. The model showed good accuracy but failed in the loss due to the low confidence of its predictions. The first problem with it is the lack of clarity over which name represents which class in the initial text, an attempt to fix this was made by adding the tags *[Name A]* and *[Name B]* to the text but the results were worse.

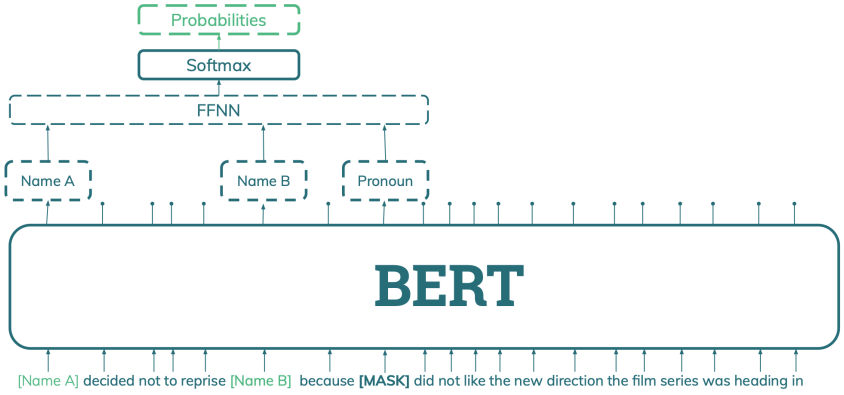

FIGURE 5.2: Model: Improved BERT for text classification

Another model that was an improved version of the BERT for text classification was then implemented (figure 5.2) to give more importance to the words involved in our problem. This so that the model wouldn't have problems identifying the names and how they are relevant to each class. This model, instead of using the results of the *[CLS* class, uses the results of the Name A, Name B and the pronoun. All three of them are passed trough a feed forward neural network and then a softmax layer. Same as the last model, it classifies in three classes A, B and None. The model was tested with and without name replacement.
The results of this second model were almost as good as the ones for Masked LM, as can be seen in table 5.8. But it was not as successful when measure by loss.

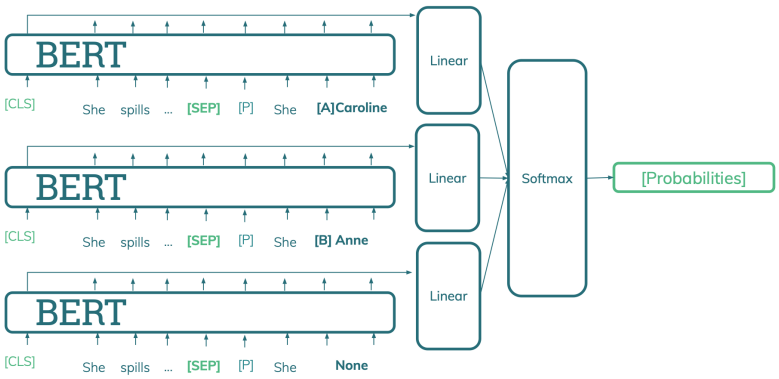

FIGURE 5.3: Model: BERT for multiple choice answering

The final of these models that were tried was a BERT for multiple choice question answering model (figure 5.3). We adapted this model so each of the text serves as the question, while the three possible answers are Name A, Name B and None (where Name A and B are the actual names, not the tags). For every answer, an entry with the format *Question [SEP] Answer* is generated. Then these entries are passed through BERT and their *[CLS]* results are added. Then these pass through a Feed Forward Neural Network and a softmax layer. This model shares some of the failings if the classification model, the distinction of the names is a little more clear but it's not enough to guarantee a good result.
Overall accuracy to the masked LM but suffered greatly with the loss. This is because in a lot of examples the difference between the probabilities of one class and another was minimal. This made for a model where each choice had low confidence and therefore the loss increased considerably.

	Accuracy	**Loss**
BERT for text classification	0.791	0.7463
Improved BERT for text classification	0.8039	0.53
BERT for Question Answering	0.731	0.8045
BERT for Masked LM Model 1	**0.835**	**0.4406**
BERT for Masked LM Model 2	0.7955	0.496

TABLE 5.8: Results of the different BERT models

5.6 Advantages of the Name Replacement

The first major advantage of the name replacement came in the architecture phase, because of the way that wordPieces split uncommon words. The tokenizer is more likely to split uncommon names and also some of the names are already multiple tokens. Because our model compares the result of one specific word (the masked pronoun) to each token then that comparison without the name replacement involves comparing the result of one token to the result of n possible tokens where n can change on every example. For our tests we decided to compare only to the first wordPiece token in each of the names. A 43% of the names across the whole GAP dataset are made up of multiple words. So replacing these with a single name makes it easier for the model to identify their place in the text. As table 4.4 shows, name replacement considerably improved the model's results. This is also because the names chosen as replacements are more common in BERT's training corpora.

	Accuracy	**Loss**
Model 1 Original Names	0.782	0.7021
Model 1 Name Replacement	0.838	0.4423

TABLE 5.9: Results for the models with and without name replacement.

In some examples the name replacement can also help with conflicting data in the text. For example the following entry:

"His maternal great-grandfather was Henry Percy, 4th Earl of Northumberland, whose wife was Maud Herbert, Countess of Northumberland. His maternal grandmother was a

daughter of Sir Robert Spencer and Eleanor Beaufort. ***[NAME A] Eleanor*** *was a daughter of Edmund Beaufort, 2nd Duke of Somerset and* ***[NAME B] Eleanor Beauchamp****.* ***[PRONOUN] She*** *was a granddaughter of Richard de Beauchamp, 13th Earl of Warwick and Elizabeth Berkeley."*

In the example above both names are share the first name Eleanor and name A has no last name to distinguish it. The name replacement makes it clearer for the model that the text is referring to different people.

5.7 Results by Gender

	Male		**Female**	
	Accuracy	**Log Loss**	**Accuracy**	**Log Loss**
Model 1	0.836	0.4173	0.84	0.4672
Model 2	0.803	0.4877	0.788	0.5043

TABLE 5.10: Results of the performance of both models divided by gender

If we take the results of both models and analyze them by gender, we can see that they share the patterns seen in the previous analysis. Model 2 had worse results than model 1 for both male and female. Results for both genders are positive and they are also very similar to each other, but female examples have slightly worse results in the loss. Better accuracy but worse loss points to a low confidence when predicting the correct result. We know the training is evenly balanced as well as the test set, so other elements could be altering the results. The names chosen for the name replacement, or BERT's pre-trained weights could have an effect in the male/female results.

Chapter 6

Conclusion

First of all, we have proven that pre-trained BERT is useful for co-reference resolution. The results for the task were positive and the model performed above our expectations. Also, this kind of solutions validate the idea behind BERT's pre-trained models. We were able to use them to create several models for a very specific task. All these models are able to be trained in a fraction of the time that it takes BERT and they are less complex than any model designed only for this task would be. This by using BERT's pre-trained models and a small additional model. After the final results for the competition were published, it can be observed that BERT was used in most of the highest ranking models, this points to its effectiveness, ease of use and also its rising popularity. It makes sense that a lot of users would be interested in trying BERT because it has been reported as the "next big thing", but the fact that the best results use it also indicates that it has a place in the modern landscape of NLP. Speaking of our project itself, it was a good way of testing out BERT's capabilities for a task that it's not designed for. Also it served as a type of exploration of BERT's different uses. From the experiments we performed we can conclude that BERT is very versatile, it can be adapted quickly to multiple tasks and it handles them well.
Looking at other results for the competition, some of the best solutions use BERT while complementing it with other models. This was missing in our approach, because our idea was to test BERT's capabilities, but for a task as specific as this one combining multiple models produced better results. These extra models helped out in the areas that BERT is lacking, like the specifics of co-reference resolution. Adding a more traditional co-reference resolution model to the solution would've helped with our model's problem with *None* values, which was the thing that harmed the results the most. Additionally, we have shown that our simple 'Name Replacement' technique was effective to reduce the impact of name frequency or popularity in the final decision. This name replacement approach was successful for this dataset, but if it's going to be used in the future it needs to be checked if the dataset is compatible. In the GAP dataset most examples work well with the name replacement, but this might not be the case in any other dataset.
A limitation of our technique is that it only covers co-reference with pronouns. A possible improvement for it would be to make the model able to process more complex co-reference entities, like gender-less objects or even phrases. Also to help with the problem of the *None* examples, it's important to analyze the characteristics of these examples where none of the names are correct and how the model could be trained better to identify them, specially because they are fewer in the dataset. It's also possible to create new *None* examples to add to the training set, to help deal with the unbalance problem. A possible improvement would be to add co-reference resolution techniques into our model, so it can help sorting out the cases where BERT is not enough. This would have to be done while considering the efficiency and resource consumption of this new model. Adding multiple models on top of each

other might work well for a competition, which is an isolated environment where you're working with one specific dataset, but when it comes to producing a version for the real world the speed and efficiency have to be taken into account.
We also have to analyze a few things about the competition and the task itself. From the gender neutrality aspect, the competition does a good job providing a challenge well suited to the objective of testing their gender neutral dataset. Looking at our results and also the top rated results in the competition, the results show parity between the genders. This means that the GAP developer's approach is a step in the right direction to solve this kind of problem. Our model and most of the highest ranking models in the competition didn't have to do any additional coding to alleviate the gender bias problem, it was done by just having a balanced dataset.

Appendix A

Short Task Paper

BERT Masked Language Modeling for Co-reference Resolution

Felipe Alfaro Lois **José A. R. Fonollosa** **Marta R. Costa-jussà**
TALP Research Center
Universitat Politècnica de Catalunya, Barcelona
felipe.alfaro@est.fib.upc.edu {jose.fonollosa,marta.ruiz}@upc.edu

Abstract

This paper explains the TALP-UPC participation for the Gendered Pronoun Resolution shared-task of the 1st ACL Workshop on Gender Bias for Natural Language Processing. We have implemented two models for mask language modeling using pre-trained BERT adjusted to work for a classification problem. The proposed solutions are based on the word probabilities of the original BERT model, but using common English names to replace the original test names.

1 Introduction

The Gendered Pronoun Resolution task is a natural language processing task whose objective is to build pronoun resolution systems that identify the correct name a pronoun refers to. It's called a co-reference resolution task. Co-reference resolution tackles the problem of different elements of a text that refer to the same thing. Like for example a pronoun and a noun, or multiple nouns that describe the same entity. There are multiple deep learning approaches to this problem. NeuralCoref [1] presents one based on giving every pair of mentions (pronoun + noun) a score to represent whether or not they refer to the same entity. In our current task, this approach is not possible, because we don't have the true information of every pair of mentions, only the two names per entry.

The current task also has to deal with the problem of gender. As the GAP researchers point out (Webster et al., 2018), the biggest and most common datasets for co-reference resolution have a bias towards male entities. For example the OntoNotes dataset, which is used for some of the most popular models, only has a 25% female representation (Pradhan and Xue, 2009). This creates a problem, because any machine learning model is only as good as its training set. Biased training sets will create biased models, and this will have repercussions on any uses the model may have.

This task provides an interesting challenge specially by the fact that it is proposed over a gender neutral dataset. In this sense, the challenge is oriented towards proposing methods that are gender-neutral and to not provide bias given that the data set does not have it.

To face this task, we propose to make use of the recent popular BERT tool (Devlin et al., 2018). BERT is a model trained for masked language modeling (LM) word prediction and sentence prediction using the transformer network (Vaswani et al., 2017). BERT also provides a group of pre-trained models for different uses, of different languages and sizes. There are implementations for it in all sorts of tasks, including text classification, question answering, multiple choice question answering, sentence tagging, among others. BERT is gaining popularity quickly in language tasks, but before this shared-task appeared, we had no awareness of its implementation in co-reference resolution. For this task, we've used an implementation that takes advantage of the masked LM which BERT is trained for and uses it for a kind of task BERT is not specifically designed for.

In this paper, we are detailing our shared-task participation, which basically includes descriptions on the use we gave to the BERT model and on our technique of 'Name Replacement' that allowed to reduce the impact of name frequency.

2 Co-reference Resolution System Description

2.1 BERT for Masked LM

This model's main objective is to predict a word that has been masked in a sentence. For this exer-

[1] https://medium.com/huggingface/state-of-the-art-neural-coreference-resolution-for-chatbots-3302365dcf30

cise that word is the pronoun whose referent we're trying to identify. This one pronoun gets replaced by the *[MASKED]* tag, the rest of the sentence is subjected to the different name change rules described in section 2.2.

The text is passed through the pre-trained BERT model. This model keeps all of its weights intact, the only changes made in training are to the network outside of the BERT model. The resulting sequence then passes through what is called the masked language modeling head. This consists of a small neural network that returns, for every word in the sequence, an array the size of the entire vocabulary with the probability for every word. The array for our masked pronoun is extracted and then from that array, we get the probabilities of three different words. These three words are : the first replaced name (name 1), the second replaced name (name 2) and the word *none* for the case of having none.

This third case is the strangest one, because the word *none* would logically not appear in the sentence. Tests were made with the original pronoun as the third option instead. But the results ended up being very similar albeit slightly worse, so the word none was kept instead. These cases where there is no true answer are the hardest ones for both of the models.

We experimented with two models.

Model 1 After the probabilities for each word are extracted, the rest is treated as a classification problem. An array is created with the probabilities of the 2 names and *none* (*[name 1, name 2, none]*), where each one represents the probability of a class in multi-class classification. This array is passed through a softmax function to adjust it to probabilities between 0 and 1 and then the log loss is calculated. A block diagram of this model can be seen in figure 1.

Model 2 This model repeats the steps of model 1 but for two different texts. These texts are mostly the same except the replacement names *name 1* and *name* 2 have been switched (as explained in the section 2.2). It calculates the probabilities for each word for each text and then takes an average of both. Then finally applies the softmax and calculates the loss with the average probability of each class across both texts. A block diagram of this model can be seen in figure 2.

2.2 Name Replacement

The task contains names of individuals who are featured in Wikipedia, and some of these names are uncommon in the English language. As part of the pre-processing for both models, these names are replaced. They are replaced with common English names in their respective genders[2]. If the pronoun is female, one of two common English female names are chosen, same thing for the male pronouns. In order to replace them in the text, the following set of rules are followed.

1. The names mentioned on the A and B columns are replaced.

2. Any other instances of the full name as it appears on the A/B columns are replaced.

3. If the name on the A/B column contains a first name and a last name. Instances of the first name are also replaced. Unless both entities share a first name, or the first name of one is contained within the other.

4. Both the name and the text are converted to lowercase

This name replacement has two major benefits. First, the more common male and female names work better with BERT because they appear more in the corpus in which it is trained on. Secondly, when the word piece encoding splits certain words the tokenizer can be configured so that our chosen names are never split. So they are single tokens (and not multiple word pieces), which helps the way the model is implemented.

Both models (1 and 2 presented in the above section) use BERT for Masked LM prediction where the mask always covers a pronoun, and because the pronoun is a single token (not split into word pieces), it's more useful to compare the masked pronoun to both names, which are also both single tokens (not multiple word pieces).

Because the chosen names are very common in the English language, BERT's previous training might contain biases towards one name or the other. This can be detrimental to this model where it has to compare between only 3 options. So the alternative is the approach in model number 2. In model 2 two texts are created. Both texts are basically the same except the names chosen as the

[2]https://www.ef.com/wwen/english-resources/english-names/

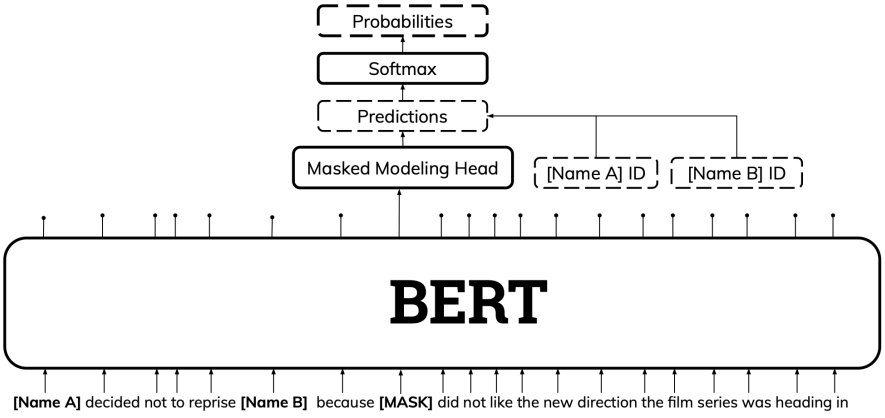

Figure 1: Model 1 representation.

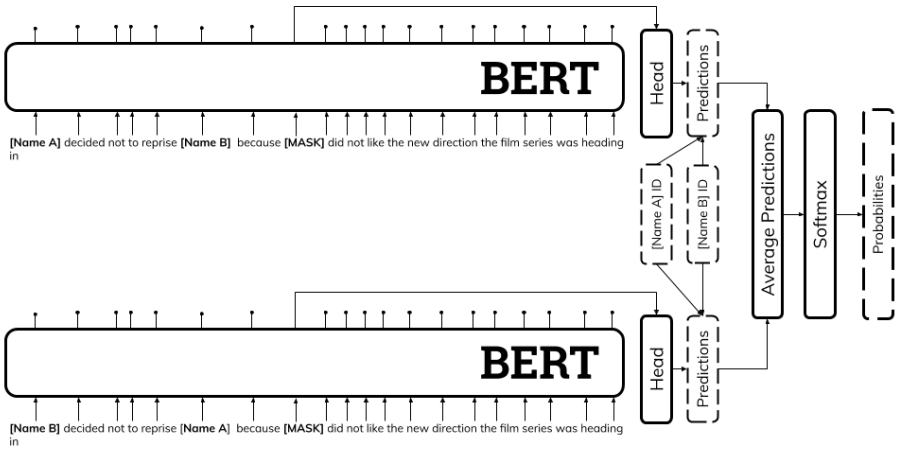

Figure 2: Model 2 representation.

Burton, who was more interested in directing Ed Wood, later reflected he was taken aback by some of the focus group meetings for **[Name B]** Forever, a title which he hated. Producer Peter MacGregor-Scott represented the studio's aim in making a film for the MTV Generation with full merchandising appeal. Production went on fast track with Rene Russo cast as Dr.ChaseMeridian but **[Name A]** decided not to reprise **[Name B]** because **[MASK]** did not like the new direction the film series was heading in.

he	**Male Name 1**	**Male Name 2**
he	**Male Name 2**	**Male Name 1**

Figure 3: Example of a text present in the dataset and how the word replacement was done for the model 2.

replacement names 1 and 2 are switched. So, as figure 3 shows, we get one text with each name in each position.

For example lets say we get the text:

"In the late 1980s Jones began working with Duran Duran on their live shows and then in the studio producing a B side single "This Is How A Road Gets Made", before being hired to record the album Liberty with producer Chris Kimsey.",

A is *Jones* and B is *Chris Kimsey*. For the name replacement lets say we choose two common English names like **John** and **Harry**. The new text produced for model 1 (figure 1) would be something like:

*"in the late 1980s **harry** began working with duran duran on their live shows and then in the studio producing a b side single "this is how a road gets made", before being hired to record the album liberty with producer **john**."*

And for model 2 (figure 2) the same text would be used for the top side and for the bottom side it would have the harry and john in the opposite positions.

3 Experimental Framework

3.1 Task details

The objective of the task is that of a classification problem. Where the output for every entry is the probability of the pronoun referencing name A, name B or Neither.

3.2 Data

The GAP dataset (Webster et al., 2018) created by Google AI Language was the dataset used for this task. This dataset consists of 8908 co-reference labeled pairs sampled from Wikipedia, also it's split perfectly between male and female representation. Each entry of the dataset consists of a short text, a pronoun that is present in the text and its offset and two different names (name A and name B) also present in the text. The pronoun refers to one of these two names and in some cases, none of them. The GAP dataset doesn't contain any neutral pronouns such as *it* or *they*.
For the two different stages of the competition different datasets were used.

- For **Stage 1** the data used for the submission is the same as the development set available in the GAP repository. The dataset used for training is the combination of the GAP validation and GAP testing sets from the repository.

- For **Stage 2** the data used for submission was only available through Kaggle[3] and the correct labels have yet to be released, so we can only analyze the final log loss of each of the models. This testing set has a total of 12359 rows, with 6499 male pronouns and 5860 female ones. For training, a combination of the GAP development, testing and validation sets was used. And, as all the GAP data, it is evenly distributed between genders.

The distributions of all the datasets are shown in table 1. It can be seen that in all cases, the *None* option has the least support by a large margin. This, added to the fact that the model naturally is better suited to identifying names rather than the absence of them, had a negative effect on the results.

[3]https://www.kaggle.com/c/gendered-pronoun-resolution/overview

	Stage 1		Stage 2
	Training Set	**Testing Set**	**Training Set**
Name A	1105	874	1979
Name B	1060	925	1985
None	289	201	490

Table 1: Dataset distribution for the datasets of stages 1 and 2.

3.3 Training details

For the BERT pre-trained weights, several models were tested. BERT base is the one that produced the best results. BERT large had great results in a lot of other implementations, but in this model it produced worse results while consuming much more resources and having a longer training time. During the experiments the model had an overfitting problem, so the learning rate was tuned as well as a warm up percentage was introduced. As table 2 shows, the optimal learning rate was $3e-5$ while the optimal with a 20% warm up. The length of the sequences is set at 256, where it fits almost every text without issues. For texts too big, the text is truncated depending on the offsets of each of the elements in order to not eliminate any of the names or the pronoun.

		Accuracy		Loss	
Learning Rate	**Warmup**	**mean**	**min**	**mean**	**min**
0.00003	**0.0**	0.840167	0.8315	0.519565	0.454253
	0.2	0.844444	0.8340	0.502667	0.442313
0.00004	**0.0**	0.822389	0.7970	0.556491	0.473528
	0.2	0.834000	0.7925	0.530862	0.456223
0.00005	**0.1**	0.743500	0.7435	0.666750	0.666750
0.00006	**0.0**	0.756333	0.7040	0.630707	0.544841
	0.2	0.802278	0.7465	0.587041	0.497051

Table 2: Results of the tuning for both models. Minimum and average Loss and Accuracy across all the tuning experiments performed.

The training was performed in a server with an Intel Dual Core processor and Nvidia Titan X GPUs, with approximately 32GB of memory. The run time varies a lot depending on the model. The average run time on the stage 1 dataset for model 1 is from 1 to 2 hours while for model 2 it has a run time of about 4 hours. For the training set for stage 2, the duration was 4 hours 37 minutes for model 1 and 8 hours 42 minutes for model 2. The final list of hyperparameters is in table 3.

Parameter	Value
Optimizer	Adam
Vocabulary Size	28996
Dropout	0.1
Sequence Length	256
Batch Size	32
Learning Rate	$3e-5$
Warm Up	20%
Steps	Stage 1: 81 — Stage 2: 148
Epochs	1
Gradient Accumulation Steps	5

Table 3: Hyperparameters for the model training

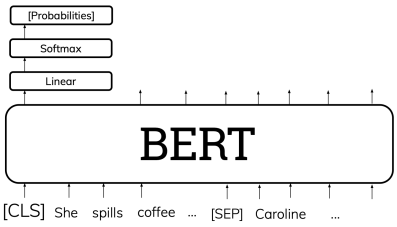

Figure 4: Model: BERT for text classification

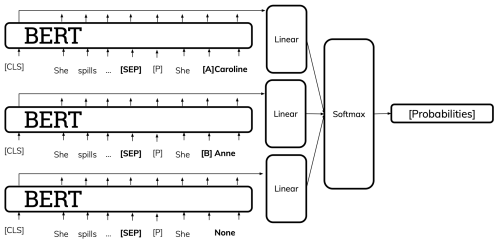

Figure 5: Model: BERT for multiple choice answering

4 Results

Tables 4 and 5 report results for models 1 and 2 reported in section 2.1 for stage 1 of the competition. Both models 1 and 2 have similar overall results. Also both models show problems with the None class, model 2 specially. We believe this is because our model is based on guessing the correct name, so the guessing of none is not as well suited to it. Also, the training set contains much less of these examples, therefore making it even harder to train for them.

	Precision	Recall	F1	Support
A	0.83	0.87	0.85	874
B	0.88	0.88	0.88	925
None	0.64	0.52	0.57	201
Avg	**0.83**	**0.84**	**0.84**	**2000**

Table 4: Model 1 results for the testing stage 1.

	Precision	Recall	F1	Support
A	0.81	0.86	0.83	874
B	0.88	0.78	0.82	925
None	0.48	0.62	0.54	201
Avg	**0.81**	**0.80**	**0.80**	**2000**

Table 5: Model 2 results for the testing stage 1.

4.1 Advantages of the Masked LM Model

As well as the Masked LM, other BERT implementations were experimented with for the task. First, a text multi class classification model (figure 4) where the *[CLS]* tag is placed at the beginning of every sentence, the text is passed through a pre-trained BERT and then the result from this label is passed through a feed forward neural network.

And a multiple choice question answering model (figure 5), where the same text with the *[CLS]* label is passed through BERT with different answers and then the result these labels is passed through a feed forward neural network.

These two models, which were specifically designed for other tasks had similar accuracy to the masked LM but suffered greatly with the log loss, which was the competition's metric. This is because in a lot of examples the difference between the probabilities of one class and another was minimal. This made for a model where each choice had low confidence and therefore the loss increased considerably.

	Accuracy	Loss
BERT for Classification	0.8055	0.70488
BERT for Question Answering	0.785	0.6782
BERT for Masked LM	*0.838*	*0.44231*

Table 6: Results for the tests with different BERT implementations.

4.2 Name Replacement Results

As table 2.2 shows, name replacement considerably improved the model's results. This is in part because the names chosen as replacements are more common in BERT's training corpora. Also, a 43% of the names across the whole GAP dataset are made up of multiple words. So replacing these with a single name makes it easier for the model

to identify their place in the text.

	Accuracy	Loss
Model 1 Original Names	0.782	0.7021
Model 1 Name Replacement	0.838	0.4423

Table 7: Results for the models with and without name replacement.

4.3 Competition results

In the official competition on Kaggle we placed 46th, with the second model having a loss around 0.301. As the results in table 8 show, the results of stage 2 were better than those of stage 1. And the second model, which had performed worse on the first stage was better in stage 2.

	Model 1	Model 2
Stage 1	0.44231	0.49607
Stage 2	0.31441	0.30151

Table 8: Results for both models across both stages of the competition

5 Conclusions

We have proved that pre-trained BERT is useful for co-reference resolution. Additionally, we have shown that our simple 'Name Replacement' technique was effective to reduce the impact of name frequency or popularity in the final decision.

The main limitation of our technique is that it requires knowing the gender from the names and so it only makes sense for entities which have a defined gender. Our proposed model had great results when predicting the correct name but had trouble with with the *none* option.

As a future improvement it's important to analyze the characteristics of these examples where none of the names are correct and how the model could be trained better to identify them, specially because they are fewer in the dataset. Further improvements could be made in terms of fine-tuning the weights in the actual BERT model.

Acknowledgements

This work is also supported in part by the Spanish Ministerio de Economía y Competitividad, the European Regional Development Fund and the Agencia Estatal de Investigación, through the postdoctoral senior grant Ramón y Cajal, contract TEC2015-69266-P (MINECO/FEDER,EU) and contract PCIN-2017-079 (AEI/MINECO).

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sameer S. Pradhan and Nianwen Xue. 2009. OntoNotes: The 90% solution. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Tutorial Abstracts*, pages 11–12, Boulder, Colorado. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the GAP: A balanced corpus of gendered ambiguous pronouns. *Transactions of the Association for Computational Linguistics*, 6:605–617.

Bibliography

Alammar, Jay (2018a). *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*. URL: http://jalammar.github.io/illustrated-bert/.

— (2018b). *The Illustrated Transformer*. URL: http://jalammar.github.io/illustrated-transformer/ (visited on 06/07/2019).

Attree, Sandeep (2019). "Gendered Ambiguous Pronouns Shared Task: Boosting Model Confidence by Evidence Pooling". In: arXiv: 1906.00839. URL: http://arxiv.org/abs/1906.00839.

Bolukbasi, Tolga et al. (2016). "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings". In: arXiv: 1607.06520. URL: http://arxiv.org/abs/1607.06520.

Caliskan, Aylin, Joanna J. Bryson, and Arvind Narayanan (2016). "Semantics derived automatically from language corpora contain human-like biases". In: DOI: 10.1126/science.aal4230. arXiv: 1608.07187. URL: http://arxiv.org/abs/1608.07187http://dx.doi.org/10.1126/science.aal4230.

Clark, Kevin and Christopher D. Manning (2016). "Improving Coreference Resolution by Learning Entity-Level Distributed Representations". In: arXiv: 1606.01323. URL: http://arxiv.org/abs/1606.01323.

Devlin, Jacob et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805. arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

Horev, Rani (2018). *BERT Explained: State of the art language model for NLP*. URL: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

Lee, Kenton et al. (2017). "End-to-end Neural Coreference Resolution". In: *CoRR* abs/1707.07045. arXiv: 1707.07045. URL: http://arxiv.org/abs/1707.07045.

Liu, Bo (2019). "Anonymized BERT: An Augmentation Approach to the Gendered Pronoun Resolution Challenge". In: arXiv: 1905.01780. URL: http://arxiv.org/abs/1905.01780.

Pradhan, Sameer S. and Nianwen Xue (2009). "OntoNotes: The 90% Solution". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Tutorial Abstracts*. Boulder, Colorado: Association for Computational Linguistics, pp. 11–12. URL: https://www.aclweb.org/anthology/N09-4006.

Shukri H., Mohd (2019a). *How Self-Attention with Relative Position Representations works*. URL: https://medium.com/@{_}init{_}/how-self-attention-with-relative-position-representations-works-28173b8c245a (visited on 05/16/2019).

— (2019b). *Why BERT has 3 Embedding Layers and Their Implementation Details*. URL: https://medium.com/@{_}init{_}/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a (visited on 06/07/2019).

Vaswani, Ashish et al. (2017a). "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30*. Ed. by I Guyon et al. Curran Associates, Inc.,

pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Vaswani, Ashish et al. (2017b). "Attention Is All You Need". In: *CoRR* abs/1706.0. arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

Webster, Kellie et al. (2018). "Mind the GAP: A Balanced Corpus of Gendered Ambiguous Pronouns". In: *Transactions of the Association for Computational Linguistics* 6, pp. 605–617. DOI: 10.1162/tacl_a_00240. URL: https://www.aclweb.org/anthology/Q18-1042.

Webster, Kellie et al. (2019). "Gendered Ambiguous Pronouns (GAP) Shared Task at the Gender Bias in NLP Workshop 2019". In: *ACL Workshop in NLP*.

Wu, Yonghui et al. (2016). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.0. arXiv: 1609.08144. URL: http://arxiv.org/abs/1609.08144.

Zhao, Jieyu et al. (2018). "Learning Gender-Neutral Word Embeddings". In: arXiv: 1809.01496. URL: http://arxiv.org/abs/1809.01496.